In [3]:
```
import pandas as pd
import numpy as np
#import random
#from collection import Counter

#visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
#import the data
train = pd.read_csv(r'D:\label\drive2Labeled.csv')
```

In [5]:
```
train.head
```

Out[5]:
```
<bound method NDFrame.head of       ENGINE_RUN_TINE ()  ENGINE_RPM ()  VEHICL
E_SPEED ()  THROTTLE ()  \
0                        0           0.00                 0    17.647058
1                        0           0.00                 0    17.647058
2                        0           0.00                 0    17.647058
3                        0           0.00                 0    17.647058
4                        0           0.00                 0    17.647058
...                    ...            ...               ...          ...
2319                  1138         674.00                 0    16.078432
2320                  1138         692.75                 0    16.078432
2321                  1140         692.75                 0    16.078432
2322                  1140         692.75                 0    16.078432
2323                  1140         692.75                 0    16.078432

      ENGINE_LOAD ()  COOLANT_TEMPERATURE ()  LONG_TERM_FUEL_TRIM_BANK_1 ()
\
0           0.000000                       76                        -4.6875
1           0.000000                       76                        -4.6875
2           0.000000                       76                        -4.6875
```

In [6]:
```
train.shape
```

Out[6]:
```
(2324, 28)
```

In [7]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2324 entries, 0 to 2323
Data columns (total 28 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   ENGINE_RUN_TINE ()                      2324 non-null   int64  
 1   ENGINE_RPM ()                           2324 non-null   float64
 2   VEHICLE_SPEED ()                        2324 non-null   int64  
 3   THROTTLE ()                             2324 non-null   float64
 4   ENGINE_LOAD ()                          2324 non-null   float64
 5   COOLANT_TEMPERATURE ()                  2324 non-null   int64  
 6   LONG_TERM_FUEL_TRIM_BANK_1 ()           2324 non-null   float64
 7   SHORT_TERM_FUEL_TRIM_BANK_1 ()          2324 non-null   float64
 8   INTAKE_MANIFOLD_PRESSURE ()             2324 non-null   int64  
 9   FUEL_TANK ()                            2324 non-null   float64
 10  ABSOLUTE_THROTTLE_B ()                  2324 non-null   float64
 11  PEDAL_D ()                              2324 non-null   float64
 12  PEDAL_E ()                              2324 non-null   float64
 13  COMMANDED_THROTTLE_ACTUATOR ()          2324 non-null   float64
 14  FUEL_AIR_COMMANDED_EQUIV_RATIO ()       2324 non-null   int64  
 15  ABSOLUTE_BAROMETRIC_PRESSURE ()         2324 non-null   int64  
 16  RELATIVE_THROTTLE_POSITION ()           2324 non-null   float64
 17  INTAKE_AIR_TEMP ()                      2324 non-null   int64  
 18  TIMING_ADVANCE ()                       2324 non-null   int64  
 19  CATALYST_TEMPERATURE_BANK1_SENSOR1 ()   2324 non-null   float64
 20  CATALYST_TEMPERATURE_BANK1_SENSOR2 ()   2324 non-null   float64
 21  CONTROL_MODULE_VOLTAGE ()               2324 non-null   float64
 22  COMMANDED_EVAPORATIVE_PURGE ()          2324 non-null   float64
 23  TIME_RUN_WITH_MIL_ON ()                 2324 non-null   int64  
 24  TIME_SINCE_TROUBLE_CODES_CLEARED ()     2324 non-null   int64  
 25  DISTANCE_TRAVELED_WITH_MIL_ON ()        2324 non-null   int64  
 26  WARM_UPS_SINCE_CODES_CLEARED ()         2324 non-null   int64  
 27  LABEL_()                                2324 non-null   int64  
dtypes: float64(15), int64(13)
memory usage: 508.5 KB
```

In [8]:
```
train.describe()
```

Out[8]:

	ENGINE_RUN_TINE ()	ENGINE_RPM ()	VEHICLE_SPEED ()	THROTTLE ()	ENGINE_LOAD ()	COOLANT
count	2324.000000	2324.000000	2324.000000	2324.000000	2324.000000	
mean	568.520224	1146.448042	20.253873	18.585772	34.284364	
std	330.315043	383.746431	14.334336	2.165776	12.250626	
min	0.000000	0.000000	0.000000	15.686275	0.000000	
25%	283.000000	771.750000	4.000000	16.470589	26.666666	
50%	569.000000	1117.750000	22.000000	18.823530	29.411764	
75%	854.250000	1475.000000	33.000000	20.000000	39.215687	
max	1140.000000	1943.750000	43.000000	28.235294	88.235291	

8 rows × 28 columns

In [9]:
```
#checking for null values
training_data_null_value =(train.isnull().sum())
print(training_data_null_value>0)
```

```
ENGINE_RUN_TINE ()                      False
ENGINE_RPM ()                           False
VEHICLE_SPEED ()                        False
THROTTLE ()                             False
ENGINE_LOAD ()                          False
COOLANT_TEMPERATURE ()                  False
LONG_TERM_FUEL_TRIM_BANK_1 ()           False
SHORT_TERM_FUEL_TRIM_BANK_1 ()          False
INTAKE_MANIFOLD_PRESSURE ()             False
FUEL_TANK ()                            False
ABSOLUTE_THROTTLE_B ()                  False
PEDAL_D ()                              False
PEDAL_E ()                              False
COMMANDED_THROTTLE_ACTUATOR ()          False
FUEL_AIR_COMMANDED_EQUIV_RATIO ()       False
ABSOLUTE_BAROMETRIC_PRESSURE ()         False
RELATIVE_THROTTLE_POSITION ()           False
INTAKE_AIR_TEMP ()                      False
TIMING_ADVANCE ()                       False
CATALYST_TEMPERATURE_BANK1_SENSOR1 ()   False
CATALYST_TEMPERATURE_BANK1_SENSOR2 ()   False
CONTROL_MODULE_VOLTAGE ()               False
COMMANDED_EVAPORATIVE_PURGE ()          False
TIME_RUN_WITH_MIL_ON ()                 False
TIME_SINCE_TROUBLE_CODES_CLEARED ()     False
DISTANCE_TRAVELED_WITH_MIL_ON ()        False
WARM_UPS_SINCE_CODES_CLEARED ()         False
LABEL_()                                False
dtype: bool
```

In [10]:
```
train.shape
```

Out[10]:
```
(2324, 28)
```

In [11]:
```
train1 = train.fillna(0)
train1
```

Out[11]:

	ENGINE_RUN_TINE ()	ENGINE_RPM ()	VEHICLE_SPEED ()	THROTTLE ()	ENGINE_LOAD ()	COOLANT_T
0	0	0.00	0	17.647058	0.000000	
1	0	0.00	0	17.647058	0.000000	
2	0	0.00	0	17.647058	0.000000	
3	0	0.00	0	17.647058	0.000000	
4	0	0.00	0	17.647058	0.000000	
...	...	...	...	...	...	
2319	1138	674.00	0	16.078432	27.843138	
2320	1138	692.75	0	16.078432	28.235294	
2321	1140	692.75	0	16.078432	28.235294	
2322	1140	692.75	0	16.078432	28.235294	
2323	1140	692.75	0	16.078432	28.235294	

2324 rows × 28 columns

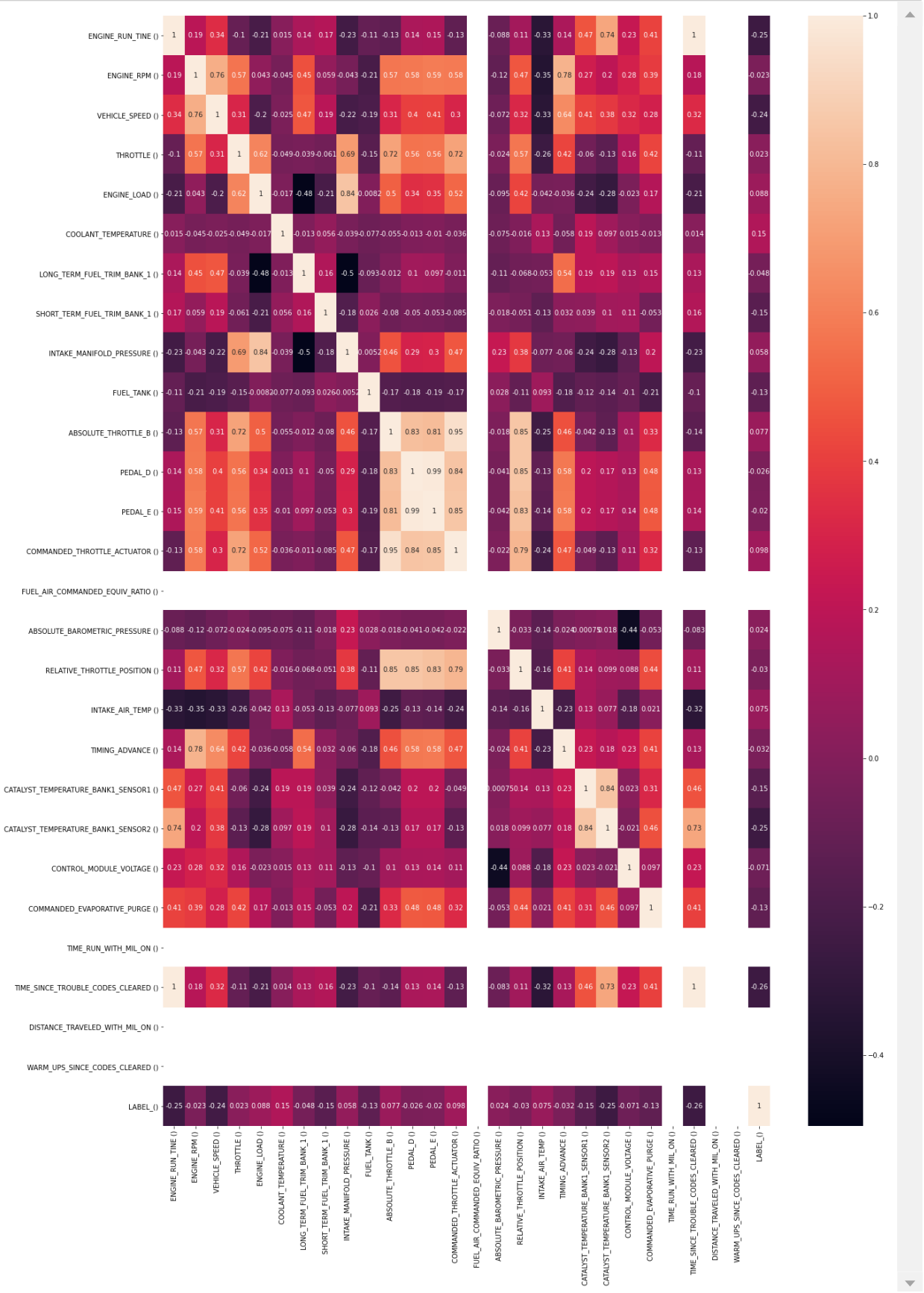
In [12]:
```
##checking for null values again
training_data_null_value =(train1.isnull().sum())
print(training_data_null_value>0)
```

```
ENGINE_RUN_TINE ()                          False
ENGINE_RPM ()                               False
VEHICLE_SPEED ()                            False
THROTTLE ()                                 False
ENGINE_LOAD ()                              False
COOLANT_TEMPERATURE ()                      False
LONG_TERM_FUEL_TRIM_BANK_1 ()               False
SHORT_TERM_FUEL_TRIM_BANK_1 ()              False
INTAKE_MANIFOLD_PRESSURE ()                 False
FUEL_TANK ()                                False
ABSOLUTE_THROTTLE_B ()                      False
PEDAL_D ()                                  False
PEDAL_E ()                                  False
COMMANDED_THROTTLE_ACTUATOR ()              False
FUEL_AIR_COMMANDED_EQUIV_RATIO ()           False
ABSOLUTE_BAROMETRIC_PRESSURE ()             False
RELATIVE_THROTTLE_POSITION ()               False
INTAKE_AIR_TEMP ()                          False
TIMING_ADVANCE ()                           False
CATALYST_TEMPERATURE_BANK1_SENSOR1 ()       False
CATALYST_TEMPERATURE_BANK1_SENSOR2 ()       False
CONTROL_MODULE_VOLTAGE ()                   False
COMMANDED_EVAPORATIVE_PURGE ()              False
TIME_RUN_WITH_MIL_ON ()                     False
TIME_SINCE_TROUBLE_CODES_CLEARED ()         False
DISTANCE_TRAVELED_WITH_MIL_ON ()            False
WARM_UPS_SINCE_CODES_CLEARED ()             False
LABEL_()                                    False
dtype: bool
```

In [11]:
```
sns.pairplot(train1)
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x22a048e5e20>
```

In [12]:
```
#creating the hot map
correlation_matrix =train1.corr()
top_correlation_feature = correlation_matrix.index
plt.figure(figsize = (20,30))
g= sns.heatmap(train[top_correlation_feature].corr(),annot =True)
```



In [13]:
```
train1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2324 entries, 0 to 2323
Data columns (total 28 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   ENGINE_RUN_TINE ()                     2324 non-null   int64  
 1   ENGINE_RPM ()                          2324 non-null   float64
 2   VEHICLE_SPEED ()                       2324 non-null   int64  
 3   THROTTLE ()                            2324 non-null   float64
 4   ENGINE_LOAD ()                         2324 non-null   float64
 5   COOLANT_TEMPERATURE ()                 2324 non-null   int64  
 6   LONG_TERM_FUEL_TRIM_BANK_1 ()          2324 non-null   float64
 7   SHORT_TERM_FUEL_TRIM_BANK_1 ()         2324 non-null   float64
 8   INTAKE_MANIFOLD_PRESSURE ()            2324 non-null   int64  
 9   FUEL_TANK ()                           2324 non-null   float64
 10  ABSOLUTE_THROTTLE_B ()                 2324 non-null   float64
 11  PEDAL_D ()                             2324 non-null   float64
 12  PEDAL_E ()                             2324 non-null   float64
 13  COMMANDED_THROTTLE_ACTUATOR ()         2324 non-null   float64
 14  FUEL_AIR_COMMANDED_EQUIV_RATIO ()      2324 non-null   int64  
 15  ABSOLUTE_BAROMETRIC_PRESSURE ()        2324 non-null   int64  
 16  RELATIVE_THROTTLE_POSITION ()          2324 non-null   float64
 17  INTAKE_AIR_TEMP ()                     2324 non-null   int64  
 18  TIMING_ADVANCE ()                      2324 non-null   int64  
 19  CATALYST_TEMPERATURE_BANK1_SENSOR1 ()  2324 non-null   float64
 20  CATALYST_TEMPERATURE_BANK1_SENSOR2 ()  2324 non-null   float64
 21  CONTROL_MODULE_VOLTAGE ()              2324 non-null   float64
 22  COMMANDED_EVAPORATIVE_PURGE ()         2324 non-null   float64
 23  TIME_RUN_WITH_MIL_ON ()                2324 non-null   int64  
 24  TIME_SINCE_TROUBLE_CODES_CLEARED ()    2324 non-null   int64  
 25  DISTANCE_TRAVELED_WITH_MIL_ON ()       2324 non-null   int64  
 26  WARM_UPS_SINCE_CODES_CLEARED ()        2324 non-null   int64  
 27  LABEL_()                               2324 non-null   int64  
dtypes: float64(15), int64(13)
memory usage: 508.5 KB
```

In [14]:
```
#independent and depentent features
X = train1.iloc[:,3:]
y = train1.iloc[:,0:2]
```

In [15]:
```
X.head(2627)
```

Out[15]:

	THROTTLE ()	ENGINE_LOAD ()	COOLANT_TEMPERATURE ()	LONG_TERM_FUEL_TRIM_BANK_1 ()	SH
0	17.647058	0.000000	76	-4.6875	
1	17.647058	0.000000	76	-4.6875	
2	17.647058	0.000000	76	-4.6875	
3	17.647058	0.000000	76	-4.6875	
4	17.647058	0.000000	76	-4.6875	
...	...	...	...	...	
2319	16.078432	27.843138	89	-1.5625	
2320	16.078432	28.235294	89	-1.5625	
2321	16.078432	28.235294	89	-1.5625	
2322	16.078432	28.235294	89	-1.5625	
2323	16.078432	28.235294	89	-1.5625	

2324 rows × 25 columns

In [16]:
```
y.head(2627)
```

Out[16]:

	ENGINE_RUN_TINE ()	ENGINE_RPM ()
0	0	0.00
1	0	0.00
2	0	0.00
3	0	0.00
4	0	0.00
...	...	...
2319	1138	674.00
2320	1138	692.75
2321	1140	692.75
2322	1140	692.75
2323	1140	692.75

2324 rows × 2 columns

In [ ]:
```
```

In [17]:
```
### Feature Importance

from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(X,y)
```

Out[17]:
```
ExtraTreesRegressor()
```

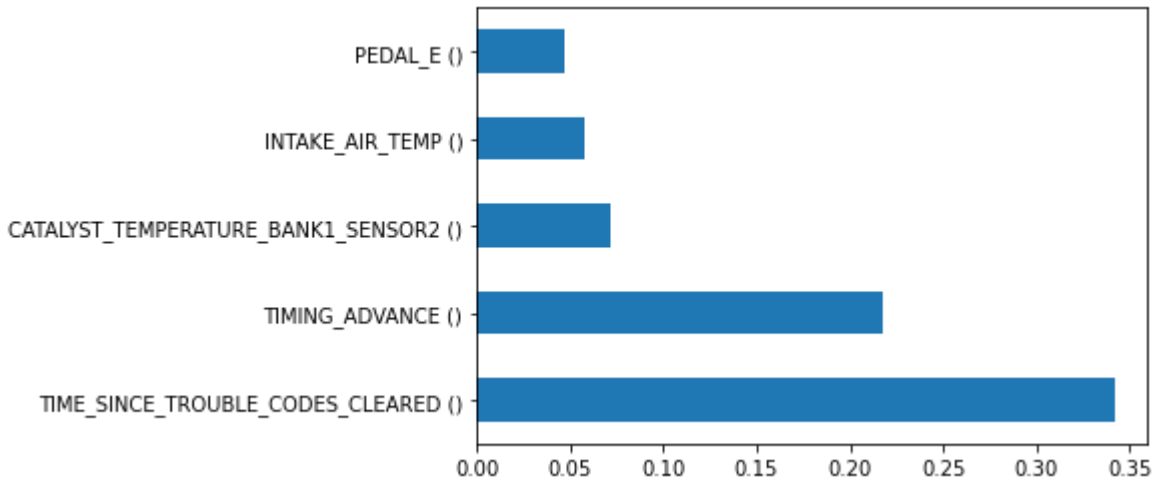
In [ ]:
```

```

In [18]:
```
print(model.feature_importances_)
```

```
[0.04229113 0.01251997 0.00843626 0.01056181 0.00512553 0.01814354
 0.00268772 0.04329176 0.02767276 0.0470232  0.02592349 0.
 0.00206077 0.01573344 0.05764437 0.21717102 0.01465951 0.07204699
 0.01482863 0.00817989 0.         0.34213523 0.         0.
 0.01186299]
```

In [19]:
```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```



In [20]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

In [45]:
```
model.score(X_train, y_train)
```

Out[45]:
```
0.9999939928382435
```

Type *Markdown* and LaTeX: α^2

In [46]:
```
model.score(X_test,y_test)
```

Out[46]:
```
0.9999986570035546
```

In [22]:
```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor()

n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
print(n_estimators)
```

```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
```

In [23]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [24]:
```
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

In [25]:
```
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

print(random_grid)
```

```
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 120
0], 'max_features': ['auto', 'sqrt'], 'max_depth': [5, 10, 15, 20, 25, 30], 'mi
n_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]}
```

In [26]:
```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
```

In [27]:
```
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
```

In [28]:
```
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   3.1s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   2.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   3.1s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   2.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   2.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   3.7s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   3.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   3.6s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   4.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   3.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   1.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   1.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   1.5s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   1.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   1.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   2.5s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   2.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   3.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   2.5s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   2.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   4.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   4.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   4.4s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   4.7s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   4.9s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
=2, n_estimators=1000; total time=   4.7s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
=2, n_estimators=1000; total time=   4.5s
```

```
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
=2, n_estimators=1000; total time=   4.2s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
=2, n_estimators=1000; total time=   4.0s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
=2, n_estimators=1000; total time=   3.9s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   2.7s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   3.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   4.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   3.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   3.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   1.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   1.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   1.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   1.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   1.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   2.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   2.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   2.6s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   2.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   2.3s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   5.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   4.9s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   4.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   4.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   4.6s
```

Out[28]:
```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
                                                         900, 1000, 1100,
                                                         1200]},
```

```
random_state=42, scoring='neg_mean_squared_error',
verbose=2)
```

In [29]:
```
rf_random.best_params_
```

Out[29]:
```
{'n_estimators': 1000,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 25}
```

In [30]:
```
rf_random.best_score_
```

Out[30]:
```
-3575.506983563281
```

In [31]:
```
predictions=rf_random.predict(X_test)
```

In [37]:

In [38]:

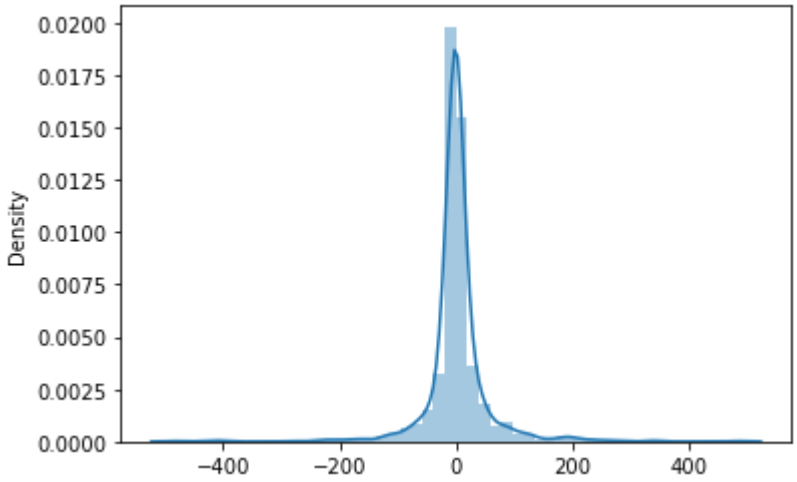
```
      ENGINE_RUN_TINE ()  ENGINE_RPM ()
2165            6.202000       38.60575
563            -1.376000        0.47175
791            -2.191000       -3.26800
1330           -5.374000      -14.82950
570            -4.383667      -18.84750
...                  ...            ...
538            -5.564000      -63.91950
1601            0.486000       -2.26125
2168           15.955000       86.51025
217             3.518250      134.39700
933             1.250000       12.56800

[698 rows x 2 columns]
```

In [32]:
```
sns.distplot(y_test-predictions)
```

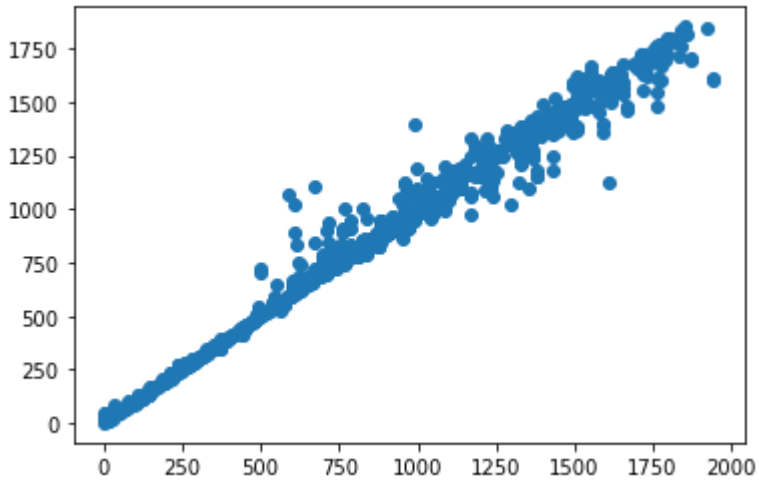
```
F:\Anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `di
stplot` is a deprecated function and will be removed in a future version. Pleas
e adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[32]: <AxesSubplot:ylabel='Density'>



In [33]:
```
plt.scatter(y_test,predictions)
```

Out[33]: <matplotlib.collections.PathCollection at 0x236bcaabaf0>



In [75]:
```
from sklearn.metrics import plot_roc_curve, plot_precision_recall_curve,roc_curve
#plot_roc_curve(rf,X_test, y_test)
```

In [60]:
```
rf.fit(X,y)
```

Out[60]:
```
RandomForestRegressor()
```

```
probs=rf.predict(X_test)
probs
```

In [70]:
```
from sklearn.preprocessing import Binarizer
```

In [71]:
```
binarizer = Binarizer(threshold=0.9)
```

In [73]:
```
s=binarizer.fit_transform(probs)
s
```

Out[73]:
```
array([[1., 1.],
       [1., 1.],
       [1., 1.],
       ...,
       [1., 1.],
       [1., 1.],
       [1., 1.]])
```

In [74]:
```
np.unique(s.ravel(), return_counts=True)
```

Out[74]:
```
(array([0., 1.]), array([   1, 1395], dtype=int64))
```

In [86]:
```
#roc_curve(X_test,y_test)
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   2.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   2.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   2.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   2.4s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split
=5, n_estimators=900; total time=   2.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   6.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   3.7s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   3.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   4.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split
=10, n_estimators=1100; total time=   5.5s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   1.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   2.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   2.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   2.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=100, n_estimators=300; total time=   1.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   3.3s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   3.3s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   3.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   3.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split
=5, n_estimators=400; total time=   3.9s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   3.9s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   4.3s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   3.9s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   4.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_spli
t=5, n_estimators=700; total time=   4.1s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
=2, n_estimators=1000; total time=   4.1s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
```

```
=2, n_estimators=1000; total time=   4.6s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
=2, n_estimators=1000; total time=   4.2s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
=2, n_estimators=1000; total time=   4.1s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split
=2, n_estimators=1000; total time=   4.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   2.7s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   2.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   2.3s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   2.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split
=15, n_estimators=1100; total time=   2.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   0.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   0.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   0.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   0.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split
=15, n_estimators=300; total time=   0.8s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   1.5s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   1.5s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   1.5s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   1.5s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=
10, n_estimators=700; total time=   1.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   4.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   4.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   4.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   4.7s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split
=15, n_estimators=700; total time=   5.3s
```

Out[86]:
```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
```

```
                                                    900, 1000, 1100,
                                                    1200]},
                                random_state=42, scoring='neg_mean_squared_error',
                                verbose=2)
```

In [90]:
```
print("Accuracy", rf_random.score(X_train,y_train))
```

```
Accuracy -418.50033421425013
```

In [91]:
```
print("Test Accuracy", rf_random.score(X_test,y_test))
```

```
Test Accuracy -3031.259373931412
```

In [87]:
```
rf_random.best_params_
```

Out[87]:
```
{'n_estimators': 1000,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 25}
```

In [89]:
```
rf_random.score(X_train,y_train)
```

Out[89]:
```
-418.50033421425013
```

```
        rf_random.
```

```

```

In [92]:
```
rf_random.return_train_score
```

Out[92]:
```
False
```

In [96]:
```
rf_random.error_score
```

Out[96]:
```
nan
```

In [98]:
```
rf_random.return_train_score
```

Out[98]:
```
False
```

In [ ]:
```
random_grid.
```